Kasu Journal of Computer Science Vol.1 No. 1 [March, 2024], pp. 1-15
https://doi.org//

Online ISSN:
Print ISSN:
www.kjcs.com.ng

A Hybridized Deep Stacked Autoencoder Model for Botnet Detection

Juliet David Shekarau[1], Rabi Mustapha[2] and Khalid Haruna[3].
Department of Computer Science, Kaduna State University, Kaduna, Nigeria.
sjulietdavid@gmail.com, rabichubu@kasu.edu.ng, khalid.haruna@kasu.edu.ng

Abstract

As the Internet aims to integrate and connect anything at anytime, anyplace with anything and by anyone, cyberattacks develop in volume and complexity with Botnets which are used in a wide range of malicious activities such as e-mail spamming; Phishing, social engineering, and even distributed denial of service (DDoS) attack. This incessant increase of attacks necessitates the interests in detecting and preventing botnet attacks in network and internet-based systems. This study develops a hybridized Deep Stacked Autoencoder optimized with Genetic Algorithm (DSAE-GA) for the classification and identification of intrusions from the internet environment. The hybridized DSAE-GA model primarily used Principal Component Analysis (PCA) technique to select a subset of features. The DSAE was trained to learn the normal network traffic profile using the Context Computer Network Traffic (CCNT) dataset while adapting to reconstruct these points with minimal reconstruction error (RE). The design of the GA majorly focuses on the parameter optimization of the DSAE thereby enhancing the classifier results.

Keywords: *Internet-based systems; Botnet detection; Optimization; Reconstruction error*

1. Introduction

Today, the Internet presents a paradigm that aims to integrate and connect anything at anytime, anyplace with anything and by anyone thus creating smart devices capable of collecting, storing, and sharing data without requiring human interaction (Bracke et al., 2021). The poor security model of these platforms and devices can be exploited by the adversary to carry out malicious and illegal actions of high-level damages, according to Li et al. (2021). The most insidious type of attack one finds today in any large-scale, distributed, Internet-connected network environment is the botnet (Ogino, 2019). The term "Botnet" is gotten from the word "RoBOT" and "NETwork". The main idea behind botnets was to control interactions in Internet Relay Chat (IRC) rooms. Bots were able to interpret simple commands, provide administration support, offer simple games and other services to chat with users and retrieve information about operating systems, login, and email address among others (Hosseini et al., 2021). Traditionally, bots' programs are constructed as clients who communicate via existing servers (Khaliq et al., 2022). This allows the bot herder (the controller of the botnet) to perform all control from a remote location, which obfuscates the traffic. Many recent botnets now rely on existing peer-to-peer networks to communicate. These peer-to-peer (P2P) bot programs perform the same actions as the client–server model, but they do not require a central server to communicate (Beauchaine et al., 2021).

Detecting Botnets is crucial to safeguard user data and ensure the smooth operation of computer systems (Owen et al., 2022). Previous research uses machine learning algorithms like Support Vector Machine

Online ISSN:
Print ISSN:
www.kjcs.com.ng

(SVM) (Masoudi-Sobhanzadeh & Emami-Moghaddam, 2022), Decision Tree (Raghavendra & Chen, 2022), Naïve Bayes (Panigrahi et al., 2022) and Random Forest (Akash et al., 2022) for detecting Botnets with promising results. However, these approaches have shown potential but they are not without challenges and limitations (Srinivasan & Deepalakshmi, 2023). Some of the problems associated with using these machine learning algorithms for Botnet detection include: limited feature extraction (Mubarak et al., 2022), inability to handle complex data patterns (Shu & Ye, 2023), scalability issues (Salman et al., 2023), limited adaptability to evolving botnet techniques (Alahmadi et al., 2023), imbalanced data distribution (Alahmadi et al., 2023), overfitting and underfitting (Alahmadi et al., 2023).

Therefore, given these problems, this study aims to detect botnet using a Deep Stacked Auto-encoder (DSAE) optimized with Genetic Algorithm (GA) as a proposed technique to engage in detecting and preventing botnet attacks in internet-based systems potentially achieving higher accuracy and adaptability in identifying Botnet activities (Soni et al., 2023).

2. Related Works

Hwang et al. (2019) applied Long Short-Term Memory (LSTM) in detecting Botnets in a Network. The novel words embedding mechanism extracts packet semantic meanings and adopts LSTM to learn the temporal relation among fields in the packet header and for further classifying whether an incoming packet is normal or a part of malicious traffic. The evaluation results on ISCX2012, USTC-TFC2016 IoT dataset. Hosseini et al. (2021) applied a Convolutional Neural Network (CNN) to detect Botnets. The model was used to compare four attacks, the IRC, Hyper Text Transfer Protocol (HTTP), Domain Name Server (DNS) and P2P, which are used by botnet. The model used network nerves and correlation with negative selection algorithm which is based on the artificial immune system to identify botnet and compared the results with random forest, K-neighbours (KNN), Support Vector Machine (SVM), Gaussian Naïve Bayes (GNB), CNN, LSTM algorithms. Karaca and Cetin (2021) proposed a Bot detection system using CNN in the Internet of Things (IoT) Environment. The developed CNN model was optimized by fluctuating the values of the hyperparameters at each repeating run. After the optimization process, the model was evaluated and obtained results were compared with GNB, Artificial Neural Network (ANN), SVM and KNN models. The proposed model performed better than GNB, ANN, SVM and KNN models with a success rate (accuracy) of 97.98%. Yerima et al. (2021) applied CNN, Deep Neural Network (DNN), LSTM, Gated Recurrent Unit (GRU), CNN-LSTM, and CNN-GRU models using 342 static features derived from the applications for Botnet detections in Android. The deep learning models achieved state-of-the-art results based on the ISCX botnet dataset and also outperformed the classical machine learning classifiers. Popoola et al. (2021) proposed a Stacked Recurrent Neural Network (SRNN) for botnet detection in smart homes. The model correctly classifies network traffic samples in the minority classes of highly imbalanced network traffic data. Multiple layers of Recurrent Neural Network (RNN) are stacked to learn the hierarchical representations of highly imbalanced network traffic data with different levels of abstraction. SRNN model with Bot-IoT dataset results show that SRNN outperformed RNN in all classification scenarios. Specifically, SRNN model learned the discriminating features of highly imbalanced network traffic samples in the training set with better

Online ISSN:
Print ISSN:
www.kjcs.com.ng

representations than RNN model. Also, SRNN model is more robust and it demonstrated better capability to effectively handle over-fitting problem than RNN model. Furthermore, SRNN model achieved better generalization ability in detecting network traffic samples of the minority classes. Apostol et al. (2021) used an Autoencoder Neural Network to detect Botnet. The technique applied an unsupervised deep learning technique to identify IoT botnet activities. An empirical evaluation of the proposed method was conducted on both balanced and unbalanced datasets to assess its threat detection capability. The model showed a reduction in False-positive rate reduction and the result obtained reveal the performance of the proposed detection method over other techniques.

Catillo et al. (2023) proposed a novel IoT-driven method for learning a single Intrusion Detection System (IDS) model that can be used to monitor the traffic of all types of IoT devices. The method is based on an integrated deep autoencoder, which is a neural network that can learn to reconstruct normal traffic data. The autoencoder is trained on data from a variety of IoT devices, which allows it to learn the common patterns of traffic for these devices. The results showed that the method achieved high accuracy.

3. Methodology

In this article, the hybridized Deep stacked autoencoder model is optimized with genetic algorithm solution for the classification and identification of intrusions from the internet-based system. The presented the DSAE-GA model primarily performs a PCA approach to select a subset of features. The DSAE was trained to learn the normal network traffic profile adapting to reconstruct these points with minimal reconstruction error (RE). The design of the GA majorly focuses on the parameter optimization of the DSAE thereby enhancing the classifier results.

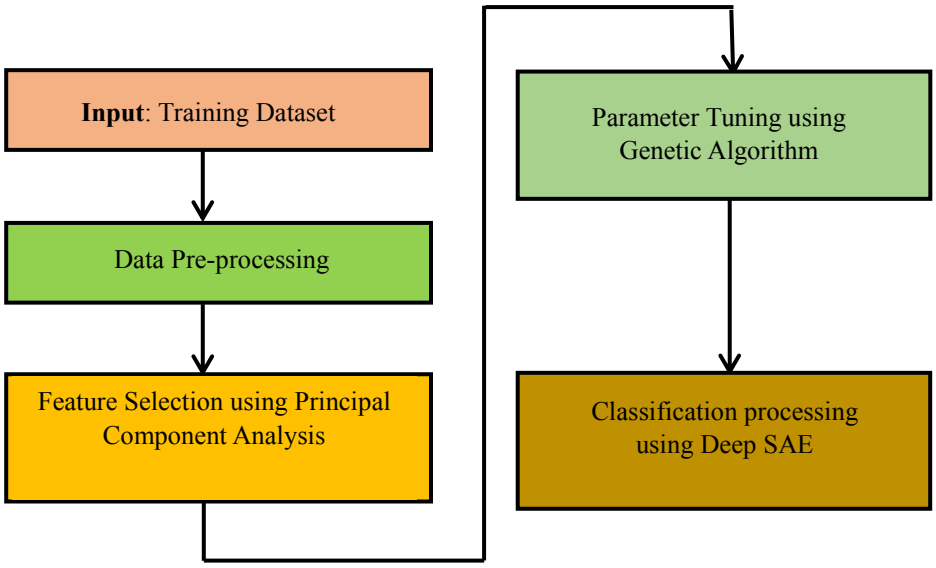


Figure 1. Block diagram of the DSAE-GA approach (Source: Duhayyim, M. A. et al., 2022)

Online ISSN:
Print ISSN:
www.kjcs.com.ng

3.1 Data Set

The dataset used is the Context Computer Network Traffic (CCNT) dataset sourced from Stanford data. This dataset encompasses network traffic information and comprises 20,803 entries, focusing on ten local workstation IP addresses. The temporal scope of this dataset spans a duration of three months.

3.2 Feature selection and extraction

The dataset underwent pre-processing to prepare it for training the botnet detection model. PCA was employed as a preprocessing technique to extract the most suitable features from the dataset.

3.3 DSAE-Based Data Classification

The DSAE with n layers is considered where each layer consists of an encoder and a decoder. The input data is denoted as X, and the hidden representations at each layer are denoted as Y^I, where n range from 1 to n. The output of the DSAE is denoted as Z.

Encoding: The encoding process in the n-th layer of the DSAE is mathematically represented in Eq. (1)

$$Y^n = f^n(W^n * Y^{n-1} + b^n) \quad (1)$$

Where Y^{n-1} is the input from the previous layer,
W^n is the weight matrix,
b^n is the bias vector,
f^n is the activation function, rectified linear unit (ReLU).

Decoding: The decoding process in the n-th layer of the DSAE is mathematically represented in Eq. (2)

$$Y'^n = g'^n(W'^n * Y'^{n-1} + b'^n) \quad (2)$$

Where Y'^{n-1} is the reconstructed input from the next layer,
W'^n is the weight matrix,
$b'^n$is the bias vector,
g'^n is the activation function.

Kasu Journal of Computer Science Vol.1 No. 1 [March, 2024], pp. 1-15
https://doi.org//
Online ISSN:
Print ISSN:
www.kjcs.com.ng

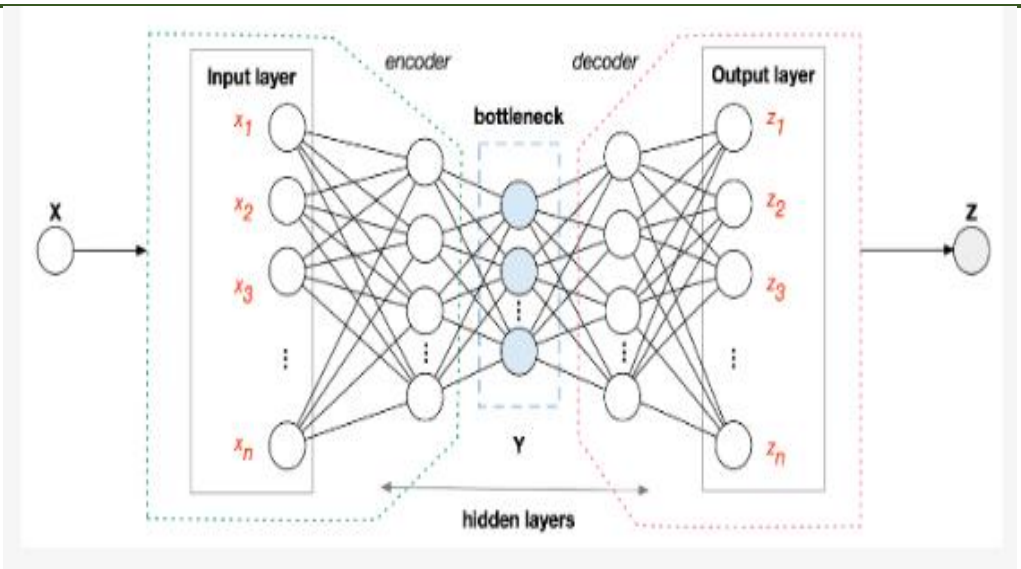


Figure 2. Structure of an Autoencoder (Source: Catillo et al., 2023)

The training process of the DSAE involves two phases: pre-training and fine-tuning;

Pre-training: Each layer of the DSAE is trained individually as a shallow auto-encoder, where the input is reconstructed by the encoder and decoder within the layer. The pre-training phase initializes the weights and biases of the DSAE.

Fine-tuning: After pre-training, the entire DSAE is fine-tuned using backpropagation and gradient descent, aiming to minimize the RE between the input and output. This phase optimizes the weights and biases of the DSAE for better performance.

Output: The output of the DSAE is obtained by passing the input through all the encoding and decoding layers in Eq. (3).

$$Z = f'^{n}(W'^{n} * Y'^{n-1} + b'^{n}) \quad (3)$$

Where f'^{n} is the activation function of the output layer,
W'^{n} is the weight matrix,
H'^{n-1} reconstructed input,
b'^{n} is the bias.

3.4 Hyperparameter Tuning Using Genetic Algorithm

Here, the GA was executed for the parameter optimization of the DSAE approach and thereby enhancing classifier results. The GA technique follows the subsequent principles:

Online ISSN:
Print ISSN:
www.kjcs.com.ng

i. **Initialization:** A population of potential solutions, often referred to as individuals or chromosomes. Each individual represented a possible solution to the problem at hand and was encoded with a set of parameters or genes.
Let P be the population and N be the population size. Each individual in the population is represented as a vector or string of genes, denoted as x. The population initialization is described in Eq. (4).

$$P = \{x1, x2, \dots, xN\} \quad (4)$$

ii. **Evaluation:** Each individual in the population was evaluated based on a fitness function that measured how well it solved the problem. The fitness function quantified the quality or performance of an individual's solution.
Let f(x) be the fitness function, where x represents an individual in the population. The fitness function maps the individual's genes or parameters to a fitness score, denoted as F(x) as shown in Eq. (5).

$$F(x) = f(x) \quad (5)$$

iii. **Selection:** Individuals were selected from the population for reproduction based on their fitness. The fitter individuals, those with higher fitness scores, had a higher probability of being selected for reproduction. This selection process mimicked the principle of "survival of the fittest" in natural evolution.
Let P(x) be the selection probability of an individual x. The selection probability is proportional to the individual's fitness value, denoted as F(x), relative to the sum of fitness values across the entire population as represented in Eq. (6).

$$P(C_x) = \left| \frac{f(C_x)}{\sum_{i=1}^{N_{Pop}} f(C_i)} \right| i = 1,2,\dots N_{pop} \quad (6)$$

Where $P(C_x)$ is probability of selecting Chromosome C_x

$f(C_x)$ is a non-negative fitness function of Chromosome C_x

$f(C_i)$ is a non-negative fitness function of Chromosome population.

iv. **Crossover:** During crossover, genetic material from two selected individuals, known as parents are combined to create an offspring. The genetic material exchange was performed at specific regions of the chromosomes. This step introduced genetic diversity and allowed the exploration of new combinations of genes.
Let P_c be the crossover probability. For each pair of selected parents, a random number R is generated. If R is less than P_c, crossover is performed; otherwise, the parents are directly copied to the next generation without any changes.

Online ISSN:
Print ISSN:
www.kjcs.com.ng

Parent 1: 1111100000

Parent 2: 0000011111

Crossover point: 5

Offspring 1: 11111 11111

Offspring 2: 00000 00000

v. **Mutation:** Mutation introduced random changes in the genetic material of individuals. It helped maintain diversity in the population and prevented the algorithm from getting stuck in local optima. Randomly selected genes within an individual's chromosome were modified or altered to introduce new traits or characteristics.
Let P_m be the mutation probability. For each gene in an individual, a random number R is generated. If R is less than P_m, the gene is mutated by applying a random modification or transformation.

Parent 1 with chromosome P1: P1_1, P1_2, ..., P1_n

Parent 2 with chromosome P2: P2_1, P2_2, ..., P2_n

Crossover point: cp

The crossover operation can be performed as follows:

Offspring 1 with chromosome O1: O1_1, O1_2, ..., O1_n

O1_i = P1_i for i <= cp

O1_i = P2_i for i > cp

Offspring 2 with chromosome O2: O2_1, O2_2, ..., O2_n

O2_i = P2_i for i <= cp

O2_i = P1_i for i > cp

vi **Termination:** The algorithm continued to iterate through the selection, crossover, and mutation steps for a predefined number of generations or until a termination condition was met. Termination conditions could include reaching a satisfactory solution, exceeding a certain computational budget, or a predetermined number of iterations.

Online ISSN:
Print ISSN:
www.kjcs.com.ng

$$Generation >= MaxGenerations \quad (7)$$

Where Generation represents the current generation number and MaxGenerations is the predefined maximum number of generations.

$$BestFitness >= Threshold \quad (8)$$

Where BestFitness represents the fitness value of the best individual in the population, and Threshold is the predefined fitness threshold.

$$ConvergenceCounter >= MaxConvergence \quad (9)$$

Where ConvergenceCounter counts the number of consecutive generations without significant improvement, and MaxConvergence is the predefined maximum number of generations for convergence.

$$TimeElapsed >= MaxTime \quad (10)$$

where TimeElapsed represents the elapsed time of the algorithm's execution, and MaxTime is the predefined maximum allowed runtime.

3.5 The Proposed Model

The DSAE-GA learns the normal network traffic profile, these DSAEs underwent training on a dataset of normal data points, adapting to reconstruct these points with minimal RE. This was achieved by comparing the RE of new data points to the RE of the normal data points. A higher RE in a new data point indicated an anomaly, leading the deep stacked auto-encoders to flag it.
The determination of whether a data point was an anomaly relied on a threshold set during the training phase. The process of establishing the threshold involved calculating the RE of the most anomalous data point within the training set.

Kasu Journal of Computer Science Vol.1 No. 1 [March, 2024], pp. 1-15
https://doi.org//
Online ISSN:
Print ISSN:
www.kjcs.com.ng

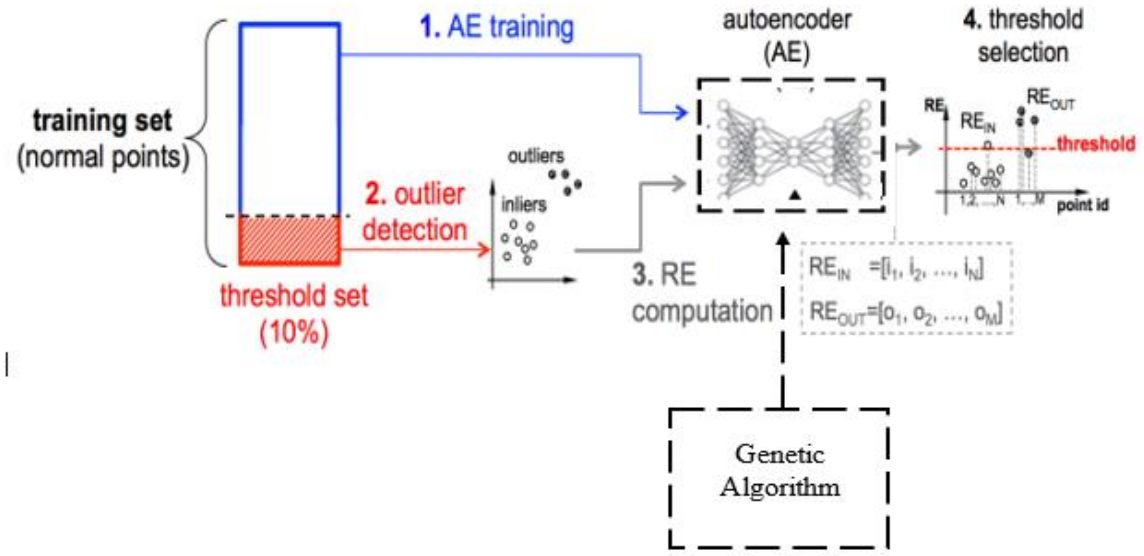


Figure 3. Threshold selection method for the proposed model (Source: Catillo et al., 2023)

In summary, the proposed method encompassed the following key stages:

i. **Deep Stacked Auto-encoder Training:** Training the deep stacked auto-encoders involves optimizing the architecture and hyper parameters of the auto-encoder to effectively learn the normal data points' profile. The Genetic Algorithm is used in tuning of the hyperparameters and architectures thereby enhancing the performance of the auto-encoder in detecting botnets.
ii. **Outlier Detection:** Utilizing an outlier detection algorithm to discern outliers within the training set.
iii. **RE Computation:** Calculating the RE for both inliers and outliers.
iv. **Threshold selection:** In this stage, the Genetic Algorithm was used to determine the appropriate threshold for botnet detection. The Genetic Algorithm optimized the threshold by considering the RE of the most anomalous outlier and conducting sensitivity analyses to strike a balance between false positives and false negatives. It explored various threshold values and evaluated their impact on the detection performance.

4. Results And Discussion

The parameters of the Deep Stacked Auto-encoder with GA Model are shown in Table 1.

Online ISSN:
Print ISSN:
www.kjcs.com.ng

Table 1: The Deep Stacked Autoencoder GA Model Parameters

Autoencoder Layer	Hidden Size	L2 Weight Regularization	Sparsity Regularization	Sparsity Proportion	Decoder Transfer Function	Softmax Layer	Deep Learning Stack
1	10	0.001	4	0.05	purelin	Loss Function	Softmax
2	10	0.001	4	0.05	purelin	crossentropy	Softmax

Table 1 presents the details of the autoencoder layers, their respective parameters, and the configurations of the deep learning stack. The first and second autoencoder layer parameters were set to the same value.

- Hidden Size: Both autoencoder layers had a hidden size of 10. This means that there were 10 neurons or units in the hidden layer of each autoencoder. The hidden size determines the capacity and complexity of the autoencoder to capture patterns and representations in the data.
- L2 Weight Regularization: The L2 weight regularization for both layers was set to 0.001. It represents the strength of the regularization or weight decay applied during training to prevent overfitting. A higher regularization value helps control overfitting but may lead to underfitting if set too high.
- Sparsity Regularization: The sparsity regularization for each layer was set to 4. It controls the strength of the regularization term that encourages sparsity in the hidden layer activations. Higher values promote sparser representations by penalizing the activation of too many neurons.
- Sparsity Proportion: The sparsity proportion for each layer was set to 0.05. It defines the desired average activation or sparsity level of the neurons in the hidden layer. A sparsity proportion of 0.05 indicates that the desired average activation of each neuron is 5%.
- Decoder Transfer Function: The decoder transfer function for both layers was set to purelin, which represents the linear transfer function. The transfer function transforms the activation values from the hidden layer to reconstruct the input data.
- The Softmax Layer: The Softmax layer was used in the deep learning stack after the autoencoder layers. It applies the softmax activation function to produce probability distributions over multiple classes. The SoftMax layer is commonly used for multi-class classification tasks.
- The training algorithm used is the genetic algorithm for hyperparameters optimization.

Kasu Journal of Computer Science Vol.1 No. 1 [March, 2024], pp. 1-15
https://doi.org//
Online ISSN:
Print ISSN:
www.kjcs.com.ng

4.1 The Parameters of the Deep Stacked Autoencoder

Figure 4.1: View of Training Parameters

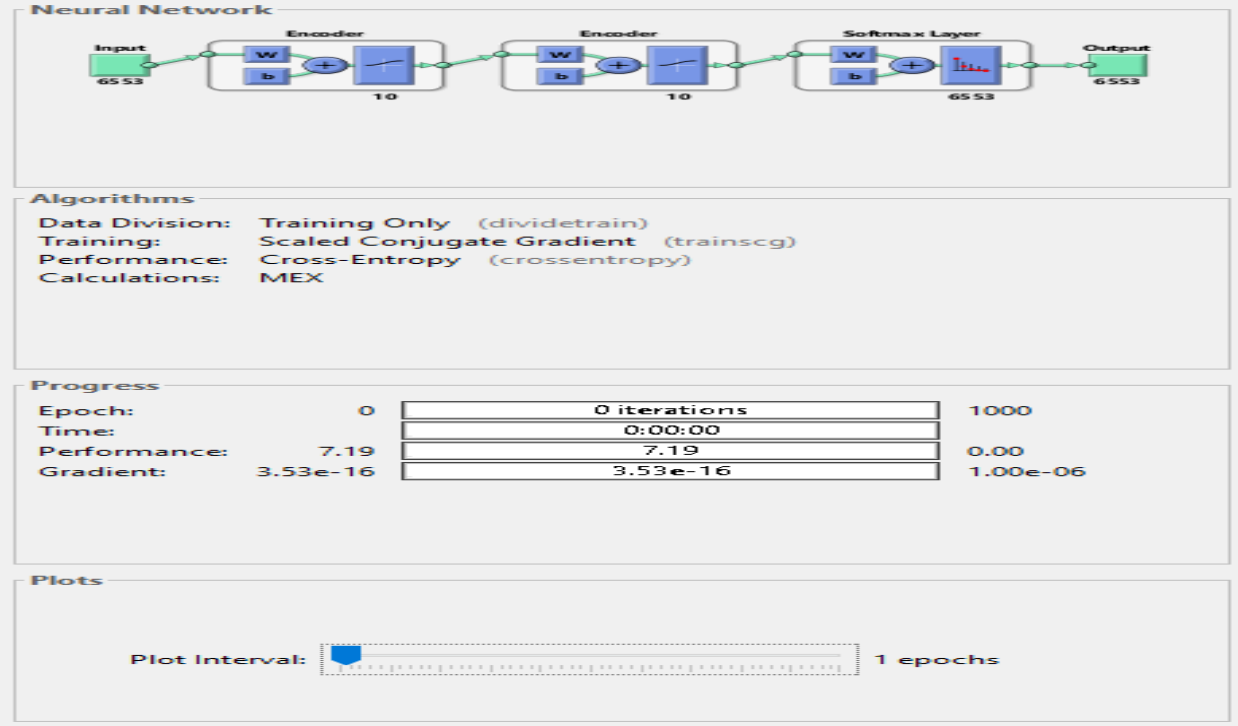
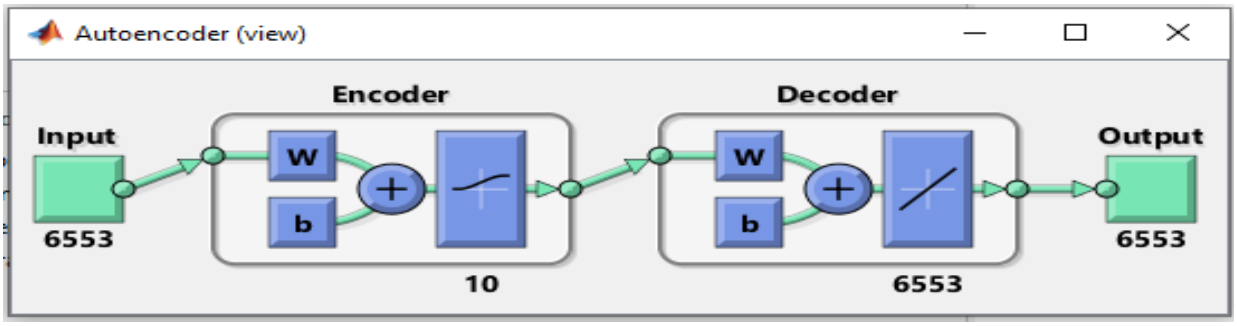


Figure 4.2: View of Network Structure



Kasu Journal of Computer Science Vol.1 No. 1 [March, 2024], pp. 1-15
https://doi.org//
Online ISSN:
Print ISSN:
www.kjcs.com.ng

Figure 4.3: View of SoftMax Classifier

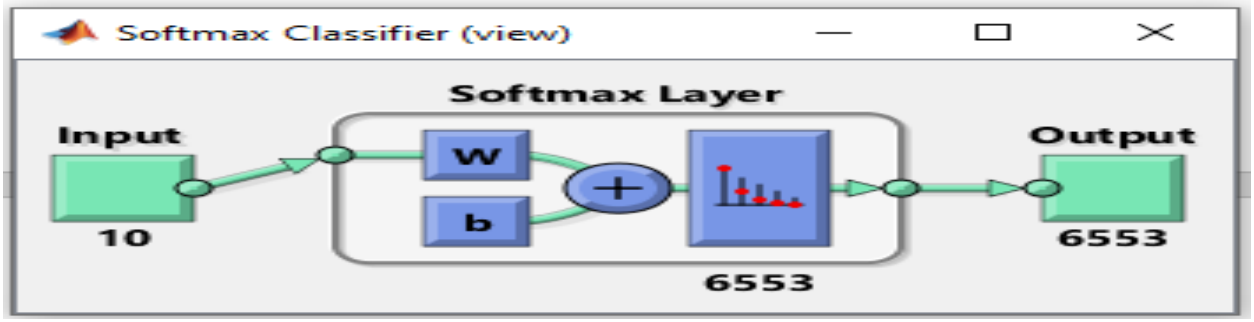
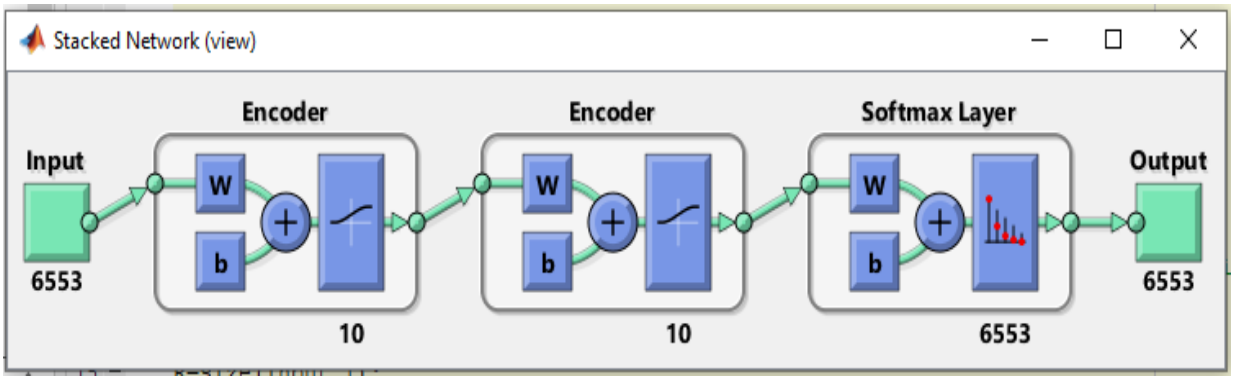
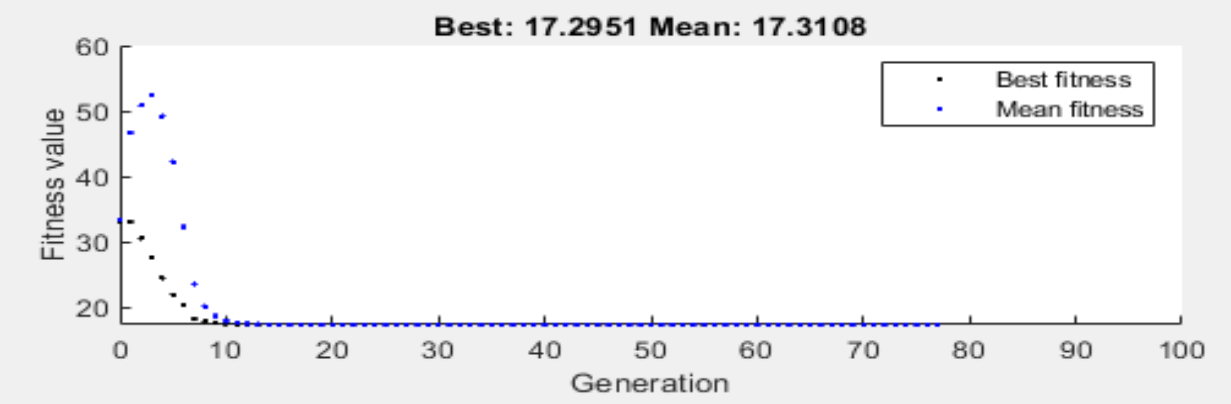


Figure 4.4: View of Stacked Network Structure



4.2 Results of GA Optimization

Figure 4.5: Training Evaluations of Genetic Algorithm



Online ISSN:
Print ISSN:
www.kjcs.com.ng

Table 2: Results of the Genetic algorithm

Model	Value
The number of generations was	77
The number of function evaluations was	14,840
The best function value found was	17.2951

The number of generations was 77 which indicates the number of iterations or generations in the Genetic Algorithm or optimization process. The number of function evaluations was 14,840. It represents the total number of times the objective function was evaluated during the optimization process. The best function value found was 17.2951. This indicates the minimum or best value obtained by the optimization algorithm for the objective function being optimized.

Table 1 provide valuable information about the configurations and performance of the autoencoder layers, the deep learning stack, and the optimization process used. The chosen parameter values reflect the design choices made to control the autoencoder's capacity, regularization, sparsity, transfer functions, and loss functions. The optimization results, such as the number of generations, function evaluations, and the best function value found, give insights into the efficiency and effectiveness of the optimization process.

5. Conclusions

In this study, an innovative DSAE-GA method was devised for the classification and identification of intrusions from the internet- based systems. The presented DSAE-GA model primarily used PCA technique which was applied to select a subset of features. The DSAE was trained to learn the normal network traffic profile adapting to reconstruct these points with minimal reconstruction error (RE). The design of the GA majorly focuses on the parameter optimization of the DSAE thereby enhancing the classifier results. Therefore, the presented DSAE-GA model was implemented as an effectual tool to recognize intrusions in the internet environment. In future, further optimization in Exploring different genetic algorithm configurations, fine
-tuning hyperparameters, and experimenting with different architectures of deep stack autoencoders could potentially enhance the performance even more. In addition, conducting robustness testing is to evaluate the proposed approach's performance under different network conditions, including variations in network traffic volume, different types of botnets, and diverse network architectures can be done in our future work.

References

Akash, N. S., Rouf, S., Jahan, S., Chowdhury, A., & Uddin, J. (2022). Botnet detection in IoT devices using random forest classifier with independent component analysis. *Journal of Information and Communication Technology*, 21. https://doi.org/10.32890/jict2022.21.2.3

Online ISSN:
Print ISSN:
www.kjcs.com.ng

Alahmadi, A. A., Aljabri, M., Alhaidari, F., Alharthi, D. J., Rayani, G. E., Marghalani, L. A., Alotaibi, O. B., & Bajandouh, S. A. (2023). DDoS attack detection in IoT-based networks using machine learning models: A survey and research directions. *Electronics*, 12(14), 3103.https://doi.org/10.3390/electronics12143103

Apostol, I., Preda, M., Nila, C., & Bica, I. (2021). IoT Botnet anomaly detection using unsupervised deep learning. *Electronics*, *10*(16), 1876. doi:10.3390/ele

Beauchaine, A., Collins, O., & Yun, M. (2021). BotsideP2P: A peer-to-peer Botnet Testbed. *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 1-4. doi:10.1109/uemcon53757.2021.9666641

Bracke, V., Sebrechts, M., Moons, B., Hoebeke, J., De Turck, F., & Volckaert, B. (2021). Design and evaluation of a scalable Internet of things backend for smart ports. *Software: Practice and Experience*, 1-5. doi:10.1002/spe.2973

Catillo, M.; Pecchia, A.;Villano, U. A Deep Learning Method for Lightweight and Cross-Device IoT Botnet Detection. Appl. Sci. **2023**, 13, 837. https://doi.org/10.3390/ app13020837

Duhayyim, M.A.; Alissa,K.A.; Alrayes, F.S.; Alotaibi, S.S.; Tag El Din, E.M.; Abdelmageed, A.A.; Yaseen, I.; Motwakel, A.Evolutionary-Based Deep Stacked Autoencoder for Intrusion Detection in a Cloud-Based Cyber-Physical System. Appl. Sci. **2022**, 12, 6875. https://doi.org/10.3390/ app12146875

Hosseini, S., Nezhad, A. E., & Seilani, H. (2021). Botnet detection using negative selection algorithm, convolution neural network and classification methods. *Evolving Systems*, *13*(1), 101-115. doi:10.1007/s12530-020-09362-1

Hosseini, S., Nezhad, A. E., & Seilani, H. (2021). Botnet detection using negative selection algorithm, convolution neural network and classification methods. *Evolving Systems*, *13*(1), 101-115. doi:10.1007/s12530-020-09362-1

Hwang, R., Peng, M., Nguyen, V., & Chang, Y. (2019). An LSTM-based deep learning approach for classifying malicious traffic at the packet level. *Applied Sciences*, *9*(16), 3414. doi:10.3390/app9163414

Karaca, K. N., & Cetin, A. (2021). undefined. *2021 International Conference on Innovations in Intelligent Systems and Applications (INISTA)*. doi:10.1109/inista52262.2021.9548445

Khaliq, Z., Khan, D. A., Baba, A. I., Ali, S., & Farooq, S. U. (2022). Model-Based Framework for exploiting sensors of IoT devices using a Botnet: A case study with Android. *arXiv*, 1-5

Li, Y., & Liu, Q. (2021). A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments. *Energy Reports*, *7*, 8176-8186. doi:10.1016/j.egyr.2021.08.126

Online ISSN:
Print ISSN:
www.kjcs.com.ng

Masoudi-Sobhanzadeh, Y., & Emami-Moghaddam, S. (2022). A real-time IoT-based botnet detection method using a novel two-step feature selection technique and the support vector machine classifier. *Computer Networks*, 217, 109365. https://doi.org/10.1016/j.comnet.2022.109365

Mubarak, A. S., Serte, S., Al□Turjman, F., Ameen, Z. S., & Ozsoz, M. (2022). Local binary pattern and deep learning feature extraction fusion for COVID□19 detection on computed tomography images. *Expert Systems*, 39(3). https://doi.org/10.1111/exsy.12842

Ogino, T. (2019). Flexible IoT edge computing system to solve the tradeoff of optimal route search. *Proceedings of the 4th International Conference on Internet of Things, Big Data and Security*, 1-4. doi:10.5220/0007587702150222

Owen, H., Zarrin, J., & Pour, S. M. (2022). A survey on Botnets, issues, threats, methods, detection and prevention. *Journal of Cybersecurity and Privacy*, 2(1), 74-88. https://doi.org/10.3390/jcp2010006

Panigrahi, R., Borah, S., Pramanik, M., Bhoi, A. K., Barsocchi, P., Nayak, S. R., & Alnumay, W. (2022). Intrusion detection in cyber–physical environment using hybrid naive Bayes—Decision table and multi-objective evolutionary feature selection. *Computer Communications*, 188, 133-144. https://doi.org/10.1016/j.comcom.2022.03.009

Popoola, S. I., Adebisi, B., Hammoudeh, M., Gacanin, H., & Gui, G. (2021). Stacked recurrent neural network for botnet detection in smart homes. *Computers & Electrical Engineering*, *92*, 107039. doi: 10.1016/j.compeleceng.2021.107039

Raghavendra, M., & Chen, Z. (2022). *Detecting IoT Botnets on IoT edge devices*. 2022 IEEE International Conference on Communications Workshops (ICC Workshops). https://doi.org/10.1109/iccworkshops53468.2022.9814555

Salman, S. A., Dheyab, S. A., Salih, Q. M., & Hammood, W. A. (2023). Parallel machine learning algorithm. *Mesopotamian Journal of Big Data*, 13-17. https://doi.org/10.58496/mjbd/2023/002

Shu, X., & Ye, Y. (2023). Knowledge discovery: Methods from data mining and machine learning. *Social Science Research*, 110, 102817. https://doi.org/10.1016/j.ssresearch.2022.102817

Soni, T., Kaur, R., Gupta, D., Sharma, A., & Gupta, G. (2023). *The cybersecurity ecosystem: Challenges, risk and emerging technologies*. 2023 7th International Conference on Trends in Electronics and Informatics (ICOEI). https://doi.org/10.1109/icoei56765.2023.10125775

Srinivasan, S., & Deepalakshmi, P. (2023). Enhancing the security in cyber-world by detecting the botnets using ensemble classification-based machine learning. Measurement: *Sensors*, 25, 100624. https://doi.org/10.1016/j.measen.2022.100624

Yerima, S. Y., Alzaylaee, M. K., Shajan, A., & Vinod, P. (2021). Deep learning techniques for Android Botnet detection. *Electronics*, *10*(4), 519. doi:10.3390/electronics10040519